TRS-80™ MODEL I/III

RSCOBOL SYSTEM USER'S GUIDE

General Information, Compiler Use, Start-Up, Sample Programs, and Sample Session

Radio Shack TRS-80 SOFTWARE™

CUSTOM MANUFACTURED IN THE USA FOR RADIO SHACK TC A DIVISION OF TANDY CORP.

IMPORTANT NOTICE

ALL RADIO SHACK COMPUTER PROGRAMS ARE LICENSED ON AN "AS IS" BASIS WITHOUT WARRANTY.

Radio Shack shall have no liability or responsibility to customer or any other person or entity with respect to any liability, loss or damage caused or alleged to be caused directly or indirectly by computer equipment or programs sold by Radio Shack, including but not limited to any interruption of service, loss of business or anticipatory profits or consequential damages resulting from the use or operation of such computer or computer programs.

NOTE: Good data processing procedure dictates that the user test the program, run and test sample sets of data, and run the system in parallel with the system previously in use for a period of time adequate to insure that results of operation of the computer or program are satisfactory.

RADIO SHACK SOFTWARE LICENSE

A. Radio Shack grants to CUSTOMER a non-exclusive, paid up license to use on CUSTOMER'S computer the Radio Shack computer software received. Title to the media on which the software is recorded (cassette and/or disk) or stored (ROM) is transferred to the CUSTOMER, but not title to the software.

B. In consideration for this license, CUSTOMER shall not reproduce copies of Radio Shack software except to reproduce the number of copies required for use on CUSTOMER'S computer (if the software allows a backup copy to be made), and shall include Radio Shack's copyright notice on all copies of software reproduced in whole or in part.

C. CUSTOMER may resell Radio Shack's system and applications software (modified or not, in whole or in part), provided CUSTOMER has purchased one copy of the software for each one resold. The provisions of this software License (paragraphs A, B, and C) shall also be applicable to third parties purchasing such software from CUSTOMER.

RADIO SHACK A DIVISION OF TANDY CORPORATION

U.S.A.: FORT WORTH, TEXAS 76102
CANADA: BARRIE, ONTARIO L4M 4W5

TANDY CORPORATION

AUSTRALIA	BELGIUM	U.K.
280-316 VICTORIA ROAD RYDALMERE, N.S.W. 2116	PARC INDUSTRIEL DE NANINNE 5140 NANINNE	BILSTON ROAD WEDNESBURY WEST MIDLANDS WS10 7JN

PRINTED IN U.S.A.

Overview of the Model I/III COBOL Documentation Package.

This binder contains the information you need to use the Radio Shack COBOL system. It assumes you are familiar with the general operation of the Computer, including use of the TRSDOS operating system. The COBOL package is provided on two Model I diskettes, the Development diskette and the Runtime diskette. Model III users will have to use the CONVERT utility to copy the COBOL package to Model III diskettes.

The COBOL system requires a minimal system of 48K RAM and two diskette drives.

The package includes three manuals.

System User's Guide

Provides general information, start-up procedures, compiler commands, creation and use of a minimal-system runtime diskette, sample programs, and a sample session. Also included is a sample session and a description of the conversion procedure for Model III users.

CEDIT User's Guide

Describes how to create and edit COBOL source files, using the COBOL editor CEDIT, which is supplied on the Development diskette.

RSCOBOL Language Reference Manual

A complete description of the Radio Shack version of the COBOL programming language. Newcomers to COBOL should consult a standard COBOL textbook for tutorial material.

```
* * * * * * * * * * * * * * * * * * * *
*                                       *
*          ALL USERS MODELS I/III       *
*   IMPORTANT NOTICE PLEASE READ FIRST  *
*                                       *
* * * * * * * * * * * * * * * * * * * *
```

Make sure you read the indicated pages for the stock number of the package that you are going to use.

STOCK NUMBER	ADDENDUM PAGES TO READ
26-2013	MODEL I version pages 1, 3, 4, 5, 6, and 7 MODEL III version page 2
26-2203	MODEL I version pages 1, 3, 4, 5, and 6 MODEL III version page 2
26-2204	MODEL I version pages 1, 3, 4, 5, and 6 MODEL III version page 2
26-2206	MODEL I pages 1, 3, 4, 5, and 6
26-2207	MODEL III page 2
26-2208	MODEL I pages 1, 3, 4, 5, and 6
26-2209	MODEL III page 2
26-1149	MODEL I version page 1, 3, 4, 5, 6, and 8 MODEL III version page 2, 8

8759170

```
* * * * * * * * * * * * * * * * * * * *
*                                       *
*            MODEL I USERS              *
*   IMPORTANT NOTICE PLEASE READ FIRST  *
*                                       *
* * * * * * * * * * * * * * * * * * * *
```

UPGRADE UTILITY ON TRSDOS 2.3B

==========

The MODEL I diskette in this package contains a NEW version of TRSDOS which is not compatible with OLD versions of TRSDOS, see below for further details.

==========

OLD TRSDOS diskettes to be used under the NEW TRSDOS MUST be UPGRADEd before use. Once UPGRADEd, a system or data diskette becomes a NEW TRSDOS data diskette.

OLD diskettes used under NEW TRSDOS without UPGRADEing, may cause extraneous information to be read at the end of files, giving a false End Of File (EOF) indication. Some programs will not function properly under these conditions.

NEW diskettes usel under OLD TRSDOS, may not access all data and/or NEW programs may not run correctly.

If you determine that you need to use the UPGRADE utility see page titled "TIPS ON USING THE MODEL I TRSDOS 2.3B UPGRADE UTILITY" contained in this addendum.

NOTE: When changing from one TRSDOS to the other you must use the RESET switch each time the diskette in drive 0 is changed.

RADIO SHACK APPLICATION PROGRAMS WHICH WERE DELIVERED ON AN OLD TRSDOS DISKETTE SHOULD NOT BE UPGRADED.

OLD:	TRSDOS 2.1, 2.2, and 2.3.
NEW:	TRSDOS 2.3B.
file:	A collection of information stored as one named unit in the directory.
program:	A file which causes the computer to perform a function.
data:	Information contained in a file which is used by a program.
system diskette:	A diskette containing TRSDOS. When this diskette is placed in drive 0 and the RESET switch is pressed, TRSDOS will begin to run.
data diskette:	A diskette which does not contain TRSDOS. If this diskette is placed in drive 0 and the RESET switch is pressed, the screen will clear and "NO SYSTEM" will be displayed.
UPGRADE:	A program contained on the TRSDOS 2.3B diskette.

```
* * * * * * * * * * * * * * * * * * * *
*                                       *
*            MODEL III USERS            *
*   IMPORTANT NOTICE PLEASE READ FIRST  *
*                                       *
* * * * * * * * * * * * * * * * * * * *
```

XFERSYS UTILITY ON TRSDOS 1.3

==========

The MODEL III diskette in this package contains a NEW version of TRSDOS which is not compatible with OLD versions of TRSDOS, see below for further details.

==========

OLD TRSDOS diskettes to be used under the NEW TRSDOS MUST be XFERSYSed before use. Once XFERSYSed, an OLD TRSDOS diskette becomes a NEW TRSDOS diskette and should not be used with OLD TRSDOS again. If you started with an OLD system or data disk, the XFERSYSed diskette will be a NEW system or data diskette respectively.

OLD diskettes used under NEW TRSDOS without XFERSYSing, may cause extraneous information to be read at the end of files, giving a false End Of File (EOF) indication. Some programs will not function properly under these conditions.

NEW diskettes used under OLD TRSDOS, may not access all data and/or NEW programs may not run correctly.

If you need to use the XFERSYS utility see the TRSDOS section of your TRS-80 MODEL III Disk System Owner's Manual.

NOTE: When changing from one TRSDOS to the other you MUST use the RESET switch each time the diskette in drive 0 is changed. You may also XFERSYS onto a NEW data disk. If this is done, all system files of the system disk will be moved onto the data disk.

RADIO SHACK APPLICATION PROGRAMS WHICH WERE DELIVERED ON AN OLD TRSDOS DISKETTE SHOULD NOT BE XFERSYSD.

OLD:	TRSDOS 1.1 and 1.2.
NEW:	TRSDOS 1.3.
file:	A collection of information stored as one named unit in the directory.
program:	A file which causes the computer to perform a function.
data:	Information contained in a file which is used by a program.
system diskette:	A diskette containing TRSDOS. When this diskette is placed in drive 0 and the RESET switch is pressed, TRSDOS will begin to run.
data diskette:	A diskette which does not contain TRSDOS. If this diskette is placed in drive 0 and the RESET switch is pressed, the screen will clear and "Not a SYSTEM Disk" will be displayed.
XFERSYS:	A program contained on the TRSDOS 1.3 diskette.

TO: Owners of the Communications Package, Series I Editor Assembler, BASIC Compiler, BASIC Runtime, COBOL Compiler, and COBOL Runtime.

FROM: Radio Shack Computer Merchandising

DATE: August 18, 1981

RE: TRSDOS 2.3B for the MODEL I

Differences between TRSDOS 2.3B and TRSDOS 2.3 are:

1. Variable length records have been corrected, in all aspects.

2. In most cases, your computer will not "hang up" when you attempt use of a device which is not connected and powered up.

3. The DEVICE command has been deleted.

4. The following commands have been added:

   CLS
   This command clears the display and puts it in the 64-character mode.

   PATCH 'filespec' (ADD=aaaa,FIND=bb,CHG=cc)
   This command lets you make a change to a program file. You need to specify:

   'aaaa' - a four byte hexadecimal address specifying the memory location of the data you want to change
   'bb' - the contents of the byte you want to find and change. You can specify the contents of more than one byte.
   'cc' - the new contents to replace 'bb'

   For example:
   PATCH DUMMY/CMD (ADD=4567,FIND=CD3300,CHG=CD3B00)
   changes CD3300, which resides at memory location 4567 (HEX) in the file named DUMMY/CMD, to CD3B00.

   If this command gives you a STRING NOT FOUND error message, this means that either 'bb' does not exist, or else 'bb' crosses a sector boundary. If 'bb' crosses a sector boundary, you must patch your file one byte at a time. For example:

   PATCH DUMMY/CMD (ADD=4568,FIND=33,CHG=3B)
   replaces the contents of the second byte in the above example.

TAPE (S=source device,D=destination device)
This command transfers Z-80 machine-language programs from one device to the other. You must specify the

'source device' and 'destination device' using these abbreviations:

T - Tape
D - Disk
R - RAM (Memory)

The only valid entries of this command are:

TAPE (S=T,D=D) TAPE (S=T,D=R) TAPE (S=D,D=T)

For example

TAPE (S=D,D=T)

starts a disk-to-tape transfer. TRSDOS will prompt you for the diskette file specification and ask you to press <ENTER> when the cassette recorder is ready for recording.

CAUTION: When doing a tape-to-RAM transfer, do not use a loading address below 6000 (Hex), since this would write over TRSDOS or the tape command.

5. These commands have been slightly changed:

BACKUP now checks to see if the diskette which will be your backup copy is already formatted. If it is, BACKUP will ask you if you want to REFORMAT it.

CLOCK will no longer increment the date when the time goes beyond 23:59:59.

COPY now works with only one-drive. For example:

COPY FILE1:0 to FILE3:0

duplicates the contents of FILE1 to a file named FILE3 on the same diskette.

KILL will now allow you to kill a protected file without knowing its UPDATE or protection level. To kill this kind of file, type an exclamation mark (!) at the end of the KILL command. For example:

KILL EXAMPLE !

kills the UPDATEd or protected file named EXAMPLE. (Note the mandatory space between the file name and the exclamation mark.)

LIST only lists the printable ASCII characters.

PROT no longer allows you to use the UNLOCK parameter.

DIR is now in this format:

```
Disk Name: TRSDOS      Drive: 0      04/15/81
Filename       Attrb   LRL   #Rec   #Grn   #Ext   EOF
JOBFILE/BLD     N*X0   256      1      1      1     1
TERMINAL/V1     N*X0   256      5      2      1   126
LOADX/CMD       N*X0   256      5      2      1     0
*** 171 Free Granules ***
```

---

1. Disk name is the name which was assigned to the disk when it was formatted.
2. File Name is the name and extension which was assigned to the file when it was created. The password (if any) is not shown.
3. Attributes is a four-character field:

a. the first character is either I (Invisible file) or N (Non-invisable file)
b. the second character is S (System file) or * (User file)
c. the third character is the password protection status of the file:
    X - the file is unprotected (no password)
    A - the file has an access word but no update word
    U - the file has an update word but no access word
    B - the file has both update and access word
d. the fourth character specifies the level of access assigned to the access word:
    0 - total access
    1 - kill the file and everything listed below
    2 - rename the file and everything listed below
    3 - this designation is not used
    4 - write and everything listed below
    5 - read and everything listed below
    6 - execute only
    7 - no access

4. Number of Free Granules - how many free granules remain on the diskette.
5. Logical Record Length - the record length which was assigned to the file when it was created.
6. Number of Records - how many logical records have been written.
7. Number of Granules - how many granules have been used in that particular file.
8. Number of Extents - how many segments (contiguous blocks of up to 32 granules) of disk space are allocated to the file.
9. End of File (EOF) - shows the last byte number of the file.

TIPS ON USING THE MODEL I TRSDOS 2.3B UPGRADE UTILITY

If you determine that you need to use the UPGRADE utility then proceed as indicated below.

Insert your TRSDOS 2.3B system diskette in drive 0, press the RESET switch, and when TRSDOS READY is displayed type UPGRADE <ENTER>. Your screen will display:

```
TRSDOS DIRECTORY UPGRADE UTILITY

FOR CONVERSION OF TRSDOS 2.1, 2.2, OR 2.3 TO
TRSDOS 2.3B DIRECTORY FORMAT.

ONCE UPGRADE HAS BEEN EXECUTED, YOUR DISKETTE SHOULD
NOT BE USED UNDER TRSDOS 2.1, 2.2, OR 2.3 AGAIN.

DO YOU WISH TO CONTINUE (Y/N/Q)?
```

This means that the directory format on your TRSDOS 2.1, 2.2, or 2.3 diskette will be converted to the TRSDOS 2.3B format. Once you type Y to continue, the screen will display:

```
INSERT DISKETTE TO BE UPGRADED IN DRIVE 1.
PRESS <ENTER> WHEN READY.
```

Insert the diskette you want to convert in drive 1 and press <ENTER>. After successful conversion, the screen will display a CONVERSION COMPLETE message. If you are attempting to convert a diskette which has already been converted, the screen will display a DISKETTE IS ALREADY A 2.3B error message.

TECHNICAL NOTE

For all files indicated in the directory that have an End Of FIle (EOF) not equal to zero, UPGRADE will change the number of records to be one less than the previous record count. Note that in FILE1, the number of records indicated has been changed from 10 to 9 after UPGRADE. For FILE2 the records indicated remain the same since EOF=0.

BEFORE UPGRADE TRSDOS 2.1, 2.2, 2.3	AFTER UPGRADE TRSDOS 2.3B
FILE1 EOF=9 10 RECORDS	9 RECORDS
FILE2 EOF=0 10 RECORDS	10 RECORDS

If the TRSDOS 2.1, 2.2, or 2.3 diskette is a system diskette, part of the conversion process will prohibit accidental usage under the TRSDOS 2.1, 2.2, or 2.3 by killing the files listed below:

SYS0/SYS	SYS1/SYS	SYS2/SYS
SYS3/SYS	SYS4/SYS	SYS5/SYS
SYS6/SYS	FORMAT/CMD	BACKUP/CMD
BASICR/CMD	BASIC/CMD	

SPECIAL NOTE FOR 26-2013 MODEL I SERIES I EDITOR/ASSEMBLER

==========

The MODEL I diskette that contains your EDTASM package includes TRSDOS 2.3B which is not compatible with TRSDOS 2.1, 2.2, or 2.3. Therefore, a machine language object file created with this package file CAN NOT simply be COPYied from TRSDOS 2.3B onto a TRSDOS 2.1, 2.2, or 2.3 diskette.

See below for instructions on how to move an object file from TRSDOS 2.3B onto a TRSDOS 2.1, 2.2, or 2.3 diskette.

==========

TIPS ON GETTING OBJECT FILES FROM TRSDOS 2.3B
ONTO TRSDOS 2.1, 2.2, OR 2.3 DISKETTES

If for example, you desire to use an assembly language function written with TRSDOS 2.3B EDTASM as a "user's external subroutine" under the TRSDOS 2.3 BASIC interpreter, follow the given steps carefully:

1) Insert your TRSDOS 2.3B system diskette that contains the EDTASM package in drive 0 and press the RESET switch.

2) Use the EDTASM package to enter and assemble a routine. We have used the SHIFT routine given in Section 7 of your TRSDOS & DISK BASIC Reference Manual as an example.

   a) Save the source program using the command:
      W SHIFT/SRC:0

   b) Then assemble the source file with the command:
      A SHIFT/CMD:0

   c) Quit EDTASM with the command:
      Q

   d) At TRSDOS READY enter the command:
      LOAD SHIFT/CMD:0

3) Remove your TRSDOS 2.3B diskette.

4) Insert your TRSDOS 2.3 diskette in drive 0 and press the RESET switch.

5) At TRSDOS READY enter the command:
   DUMP SHIFT/CMD:0 (START=X'7D00',END=X'7D09',TRA=X'7D00')

   Reference Section 4 of your manual and note that X'7000' is the lowest address that may be used as the origin of your programs.

6) The file on this diskette, named SHIFT/CMD, may now be used as needed under TRSDOS 2.1, 2.2, or 2.3 with the BASIC interpreter as a user's external subroutine.

```
* * * * * * * * * * * * * * *
*                           *
*      IMPORTANT NOTICE     *
*            FOR            *
*   COMMUNICATIONS PACKAGE  *
*     DISK SYSTEM USERS     *
*                           *
* * * * * * * * * * * * * * *
```

The 26-1149 Communications Package is delivered on MODEL I TRSDOS 2.3B and Model III TRSDOS 1.3. Communication can occur MODEL I to I, III to III, or I to III, but only under MODEL I TRSDOS 2.3B and MODEL III TRSDOS 1.3.

Data on MODEL I TRSDOS 2.1, 2.2, or 2.3 must be UPGRADEd to 2.3B beofre it can be transmitted. Backup the diskette before UPGRADEING.

Data on MODEL III TRSDOS 1.1 and 1.2 must be XFERSYSed to 1.3 before it can be transmitted. Backup the diskette before XFERSYSing.

NOTE: Radio Shack Application programs on TRSDOS 1.1, 1.2, 2.1, 2.2, or 2.3 were tested on the particular version of TRSDOS they were purchased on.

No quarantee is implied that these programs will work correctly after being UPDATEd to MODEL I TRSDOS 2.3B or XFERSYSed to MODEL III TRSDOS 1.3.

IMPORTANT NOTE FOR MODEL I USERS: You cannot run BASIC programs because TRSDOS 2.3 does not contain DISK BASIC.

On page 20 of the Communications Package manual, we suggest you use SAVE, a DISK BASIC command, to save a transferred BASIC tape program on diskette. You will not be able to use the SAVE command with the TRSDOS 2.3B diskette, since it does not contain DISK BASIC.

Important Note for Model I/III RSCOBOL Users

The object modules (described on page 32 of the **Use** section) are password protected and cannot be copied (with COPY) from your system diskette. To transfer these modules to another diskette, use BACKUP.

When you assign a program-name to a COBOL file (refer to page 29 of the **RSCOBOL** section of this manual), you must use standard TRSDOS syntax for the program-name. See the **File Specification** section of your Model III owner's manual for specific details.

Thank-You!

Radio Shack®

A Division of Tandy Corporation

8759117-581

TRS-80 Model I/III

COBOL USER'S GUIDE

(RS/COBOL 1.3)

December, 1980

PREFACE

This document contains the information required to compile, run and debug COBOL language programs on the Radio Shack TRS-80 Model I/III Microcomputer under the TRSDOS Disk Operating System.

It assumes the reader is familiar with the COBOL Language, the general operation of the TRS-80 Model I or Model III Microcomputer, and the TRSDOS Operating System. The reader is specifically referred to the following publications:

TRS-80 Model I/III COBOL Language Manual
TRS-80 Model I Operation Manual
TRS-80 Model I Disk Operating System Reference Manual
TRS-80 Model III Disk Operating System Reference Manual

This guide is organized such that each chapter fully describes a particular operational procedure. While the experienced user need only refer to the appropriate chapter, it is recommended that the first-time user read the complete guide prior to operation of the COBOL system.

PROPRIETARY RIGHTS NOTICE

TRS-80 Model I/III COBOL (RSCOBOL) is a proprietary product of:

Ryan-McFarland Corporation
Software Products Group

licensed to:

Tandy Corporation
One Tandy Center
Fort Worth, Texas 76102
(817) 390-3583

The software described in this document is furnished to the user under a license for use on a single computer system and may be copied (with inclusion of the copyright notice) only in accordance with the terms of such license.

TABLE OF CONTENTS

CHAPTER 1

THE COBOL COMPILER

1.1 Compiler Overview

The COBOL Compiler operates on a 48K byte TRS-80 Model I or Model III Microcomputer with at least two disk drives under the appropriate TRSDOS Operating System. (Model I - version 2.3, Model III - version 1.1).

Once executed, the Compiler makes a single pass on the source program, generating object and listing files concurrently. Upon completion it reports compilation results on the display and returns control to TRSDOS.

Compilation always proceeds to the end of the program, regardless of the number of source errors found.

A listing of the program is generated showing the original COBOL source statements, error information, data allocation, Interactive Debug information and, optionally, a Cross Reference of all program labels and data items. This listing can be directed to the Console, the Printer and/or a disk file.

The generated object file is in a form ready for immediate execution by the COBOL Runtime. Object code is produced such that an attempt to execute an erroneous statement will terminate execution with an appropriate error message.

1.2 Device Assignments

All communication between the Compiler and the User is through the system console.

During operation, the Compiler will require one or more of the following devices:

Display & Keyboard compiler command input & compiler messages

Disk	source input file
Disk	listing file (optional)
Disk	object file (optional)
Disk	COPY input file (optional)
Display	listing display (optional)
Printer	listing print (optional)

1.3 Executing the Compiler

To compile a COBOL source program, issue the following command to TRSDOS:

RSCOBOL filespec (options) comment

where:

filespec

is the file specification of the COBOL source file to be compiled; of the form:

filename/ext.password:d

'filename' is required.

'/ext' is an optional name-extension. When omitted, the default '/CBL' is used.

'.password' is an optional password. Note: If the file was created with a nonblank password, '.password' becomes a required field.

':d' is an optional drive specification. When omitted, the system does an automatic search, starting with drive 0.

options

allows the user to specify compiler and/or file options. Each option must be specified as shown below, separated by spaces. The left and right parenthesis are required if any comments are present.

When no options are specified, the compiler will automatically generate an object file but no listing output.

1.3.1 Compiler Source Input

The Compiler expects the source input to be a sequential file, containing logical records of ASCII text. These logical records can be either of two forms; 'byte-stream' or 'fixed':

'byte-stream' records consist of a string of ASCII characters, terminated by a carriage-return character. This format is typically stored on the disk as one byte records (LRL=1), and is the format created by the standard TRSDOS editor(s).

'fixed' records consist of 80 ASCII characters each (LRL=80), and do not contain carriage-return or other special characters.

1.3.2 Compiler Options

D

'D' instructs the compiler to compile all COBOL "Debug" source lines, identified by a "D" in column 7. This allows the user selective compilation of COBOL source statements.

This option has no relationship to the COBOL Runtime Interactive Debug facility and need not be specified to allow such debugging.

The default is to treat such lines as comments.

E

'E' instructs the compiler to generate an 'Error Only' listing instead of a full listing. This option is effective only when a listing has been specified (L, P and/or T options).

The listing generated will contain the page heading information, all source lines in error with their appropriate undermarks and messages, plus all summary information.

The default is not to generate an error listing.

L L=d

'L' indicates that the compiler listing is to be written to a disk file with the name of the source file and a filename-extension of '/LST'. The first available disk is used.

Specifying a drive number (L=d) indicates that the listing file is to be written to disk 'd'.

LST files may be displayed using the standard TRSDOS LIST and PRINT utilities.

The default is not to generate a listing file.

O O=d O=N

'O' indictes that the Compiler object output is to be written to a disk file with the name of the source file and a filename-extension of '/COB'. The first available disk is used.

Specifying a drive number (O=d) indicates that the object file is to be written to disk 'd'. When omitted the first available disk is used.

'O=N' indicates that no object file is to be generated.

The default is to generate an object file on the first available disk.

P

'P' indicates that the listing is to be printed on the printer.

The default is not to print the listing.

T

'T' indicates the listing is to be displayed on the system display.

The default is not to display the listing.

X

'X' indicates a cross-reference of COBOL Procedure and Data Division names is to be produced. This option is effective only when a listing has been specified (L, P or T options).

The default is not to generate a cross-reference.

1.3.3 Compiler Messages

Messages which report the compiler's status, or its ability to complete the compilation process are reported on the system display as they are detected.

TRS-80 Model I/III COBOL Compiler (RM/COBOL ver v.r)
Copyright 1980 by Tandy Corp. Licensed from Ryan-McFarland Corp.

> Indicates that the compiler has been loaded and has begun to compile the specified program. 'ver v.r' identifies the version (v) and revision (r) level of the compiler.

COMPILATION COMPLETE: eeee ERRORS, wwww WARNINGS

> Indicates that the compilation has been completed. The values of 'eeee' and 'wwww' indicate the number of errors and warnings, respectively, identified in the source program. This message is repeated on the listing.

PARAMETER ERROR AT: vvvvvvvv

> Indicates that an unrecoverable error was detected on the command to execute the compiler. 'vvvvvvvv' will identify the offending field.
>
> The user should reenter the command with the necessary corrections.

COMPILATION CANCELLED

> Compiler cancelled by user with BREAK key.

COMPILER ERROR, NO: nnnn

An internal error has occurred which prevents continued compilation. The value of 'nnnn' identifies the condition which caused the error.

0001 Pointer overflow
The user program has exceeded internal compiler pointers. Segment the program and recompile. If this problem still exists, separate programs into main program with multiple subroutines.

0002 Roll memory overflow
The user program has exceeded available work space. Segment the program and recompile.

0010 Unable to locate or load a compiler overlay.
Install the RSCBLnvr program overlays as described in the chapter on 'Installation Procedures.'

0020 Invalid TRSDOS
Execution was attempted under an incorrect version of TRSDOS Order correct version of TRSDOS.

Required TRSDOS versions are:

Model I - 2.3
Model III - 1.1

0030 Invalid Source Record
The Compiler has encountered an invalid source input record. Verify records are ASCII text, formatted as either:

a) Variable length records (LRL=1) terminated with a carriage return, or;

b) Fixed length 80 character records (LRL=80) without carriage return.

1.3.4 Examples

```
RSCOBOL PAYROLL (P X)
```

locates and compiles the source program PAYROLL/CBL, producing an object file (PAYROLL/COB) on the first available disk and a listing, with cross-reference, on the printer.

```
RSCOBOL MORTGAGE/SRC:1 (L=2 O=N)
```

compiles the source program MORTGAGE/SRC located on the disk in drive 1, producing a listing file (MORTGAGE/LST) on the disk in drive 2, and no object file.

1.4 The Program Listing

The compiler outputs 'source', 'allocation', and 'summary' listings if a listing device or file is specified (L, P or T options). When the 'X' option is specified, a 'cross-reference' listing is also produced.

The source listing includes a sequential line number, sentence address, source image, and interspersed diagnostics.

The allocation listing includes the address, size, order, type, and name of each identifier. The identifier names are indented to show the record structure. (The order of an identifier is the number of subscripts it requires).

The summary listing includes the number of errors, the number of warnings, and the size of the program.

The cross-reference listing includes all identifier names in alphabetical order, and the line number of each declaration, source, and destination reference. The line number is surrounded by slashes if the reference is a declaration; asteriks if the reference is a possible modification. References to all paragraphs and sections are included.

In all listings, numbers in decimal are represented as ddd...d, numbers in hexadecimal are represented as >dd...d.

1.4.1 Listing Diagnostics

Source statements are checked for syntax and semantic errors as they are scanned. Errors may cause interruption in scanning. In this case, text is ignored until a recovery point is found and a resume message is printed. Recovery points are chosen to minimize the amount of unanalyzed text without producing irrelevant error messages. In any case, the constructs at fault are undermarked and error messages listed when the source line is printed. The error message includes either E's or W's indicating error or warning. For example:

```
004030  02  STOCK  PIC  9(16)PPP COMPUTATIONAL.
                                  $
***** 1)PICTURE  *E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E
```

Indicates a semantic number size error but

```
005040  02  PART  PIC X(4BX(5)     SYNC.
                        $          $
***** 1)SYNTAX *E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E
***** 2)SCAN RESUME  *W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W*W
```

indicates a syntax error at the first undermark and a recover at the second undermark.

The number preceding the error message is the undermark number, counting from left to right. More than one message may refer to the same undermark.

Global errors such as undefined paragraph names and illegal control transfers are listed with the program summary at the end of the source listing.

1.4.2 Diagnostic Messages

ACCESS CLASH
: Nonsequential access given for sequential file.

BLANK WHEN ZERO
: BLANK WHEN ZERO clause given for nonnumeric or group item.

CLASS
: The referenced identifier is not valid in a class condition.

COPY
: COPY statement failed because of permanent error associated with the undermarked file-name.

CORRESPONDING
: The CORRESPONDING phrase cannot be used with the referenced identifier.

DATA OVERFLOW
: The data area (working-storage and literals) is larger than 65535 bytes in length.

DATA TYPE
: Context does not allow data type of the referenced identifier.

DEVICE CLASH
: Random characteristics given to nonrandom device.

DEVICE TYPE
: OPEN or CLOSE mode inconsistent with device type.

DOUBLE DECLARATION
: Multiple declaration of a file or identifier attribute.

DOUBLE DEFINITION
: Multiple definition of an identifier.

DUPLICATE

Warning only. Multiple USE procedure declared for same function or file.

FILE DECL ERROR

The referenced file-name is SELECTed and has an invalid or missing file description (FD).

FILE NAME ERROR

The referenced file-name has an invalid external file name declaration.

FILE NAME REQUIRED

File name not given as referenced in I/O verb.

FILE RECORD KEY ERROR

The referenced file-name has a RECORD KEY which is incorrectly qualified or is not defined as a data item of the category alphanumeric within a record description entry associated with that file name.

FILE RECORD SIZE ERROR

The referenced file-name has a declared record size which conflicts with the actual data record descriptions or is a relative organization file with variable length records.

FILE RELATIVE KEY ERROR

The referenced file-name has a RELATIVE KEY which is incorrectly qualified, is defined in a record description associated with that file-name, or is not defined as an unsigned integer.

FILE STATUS ERROR

The referenced file-name has a status item which is incorrectly qualified, is not defined in the WORKING-STORAGE SECTION, or is not a two-character alphanumeric item.

FILE TYPE

Access or organization of file conflicts with undermarked statement.

FILLER LEVEL

A non-elementary FILLER item is declared.

GROUP CLASH

USAGE or VALUE clause of group member conflicts with same clause for group.

GROUP VALUE CLASH

Warning Only. An item subordinate to a group with the VALUE IS clause is described with the SYNCHRONIZED, JUSTIFIED, or USAGE (other than USAGE IS DISPLAY) clause.

IDENTIFIER
Identifier reference is incorrectly constructed or the identifier has an invalid or double definition.

ILLEGAL ALTER
An ALTER statement references an unalterable paragraph or violates the rules of segmentation.

ILLEGAL PERFORM
A PERFORM statement reference undefined or incorrectly qualified paragraph or the reference violates the rules of segmentation.

INVALID ID
The referenced identifier was not successfully defined.

INVALID PARAGRAPH
Context does not allow section name.

JUSTIFY
JUSTIFY clause given in conflict with other attributes.

KEY REQUIRED
Relative key not declared for random access relative file or record key not declared for indexed file.

LABEL
Presence or absence of label record conflicts with device standards.

LEVEL
Level-number given is invalid either intrinsically or because of position within a group.

LINKAGE
An identifier in the USING clause of the PROCEDURE title is not a linkage item or a statement references a linkage item not subordinate to an identifier in the USING clause of the PROCEDURE title.

LITERAL VALUE
Literal value given is incorrect in context.

MOVE
Operands of MOVE verb specify an invalid move.

MUST BE INTEGER
Context requires decimal integer.

MUST BE PROCEDURE
Context requires procedure name either as reference or definition, or the reference must be a nondeclarative procedure-name.

MUST BE SECTION
: Context requires procedure-name to be section.

NESTING
: Illegal nesting of condition that is not an IF condition.

NOT IN REDEFINE
: VALUE IS clause given in REDEFINES item.

OCCURS
: OCCURS clause given at invalid level or after three have been given for the same item.

OCCURS DEPENDING ERROR
: The referenced object of a DEPENDING phrase has not been defined correctly.

OCCURS-VALUE CLASH
: VALUE IS and OCCURS in effect for the same item.

PICTURE
: Invalid PICTURE syntax.

PICTURE-BWZ CLASH
: Zero suppression and BLANK WHEN ZERO cannot be in effect for the same item.

PICTURE-USAGE CLASH
: USAGE clause or implied usage conflicts with usage implied by picture.

PROCEDURE INDEPENDENCE
: PERFORM given for procedures in independent segments not in the current segment.

PROGRAM OVERFLOW
: The instruction area is larger than 32767 bytes in length.

RECORD KEY
: Record key declared for other than an indexed organization file or a START statement KEY phrase references a data item not aligned on the declared key's leftmost byte.

RECORD REQUIRED
: Context requires record name.

REDEFINES
: REDEFINES given within an OCCURS or not redefining the last allocated item.

REDEFINES ERROR
The referenced data-name redefines an item which does not have the same number of character positions and is not level 01.

REFERENCE INVALID
Reference given is not valid in context.

RELATION
Operands of relation test are incompatible.

RELATIVE KEY
Relative key declared for other than a relative organization file or a START statement KEY pharase references a data item other than the declared key.

RESERVED WORD CONFLICT
A COBOL reserved word or symbol is given where a user word is required. In the summary this is only a warning about an ANSI COBOL reserved word that is not an implemented COBOL reserved word.

SCAN RESUME
Warning only. Scanning was terminated at previous error message and resumes at undermarked character.

SECTION CLASH
A VALUE IS clause appears in the FILE or LINKAGE section.

SEGMENT
Warning only. Segment number given in an independent segment is not the same as the current segment or the number of a new independent segment. The current segment number is used.

SEPARATOR
Warning only. Redundant punctuation or a separator is not followed by the required space.

SIGN
SIGN clause given in conflict with usage and picture.

SIZE
Warning only. Size of data referenced not correct for context.

SIZE ERROR
Declared size of record conflicts with present reference.

SUBSCRIPT
Incorrect number of subscripts or indices for a reference.

SYNC

Synchronized clause given for a group item

SYNTAX

Incorrect character or reserved word given for context.

UNDEFINED

File referenced in FD entry was not defined.

UNDEFINED DECLARATIVE PROCEDURE

A declarative statement references a procedure not defined within the DECLARATIVES.

UNDEFINED PROCEDURE

A GO TO statement references an undefined or incorrectly qualified paragraph.

USE REQUIRED

A DECLARATIVES section must begin with a USE statement.

USING COUNT

Warning only. The item count in the USING list of a CALL statement is different from that of the first reference to the same program name.

VALUE ERROR

Value given in VALUE IS required truncation of nonzero digits.

VALUE

VALUE IS clause given in conflict with other declared attributes.

VARIABLE RECORD

Warning only. The INTO phrase is not allowed with variable size records.

CHAPTER 2

THE COBOL RUNTIME

2.1 Runtime Overview

The COBOL runtime operates on a 48K byte TRS-80 Model I or Model III Microcomputer with at least two disk drives under the appropriate TRSDOS Operating System. (Model I - version 2.3, Model III - version 1.1).

Once invoked, the runtime loads and executes the compiled object program, automatically loading any required segments. Concurrently, it allocates memory for file buffers, and CALLed COBOL and Assembly Language subprograms. Upon completion appropriate messages are displayed and control is returned to the operating system.

2.2 Device Assignments

All communication between Runtime and the User is through the keyboard and display.

During operation the Runtime will require one or more of the following devices:

Keyboard & Display	runtime command input, Interactive Debug command input, and runtime messages.
Keyboard & Display	ACCEPT and DISPLAY, and Interactive Debug display.
Printer	PRINT output, if required.

NOTE: For PRINT output, the device name "PRINTER" must be specified in the SELECT statement; i.e,

SELECT filename, ASSIGN to PRINT, "PRINTER".

2.3 Executing the Compiled Program

To execute a compiled COBOL object program, issue the following command to TRSDOS:

RUNCOBOL filespec (options) comment

where:

filespec

is the specification of the compiled COBOL object file to be executed of the form:

filename/ext.password:d

'filename' is required.

'/ext' is an optional name-extension. When omitted the default '/COB' is used.

'.password' is an optional password. Note: If the file was created with a nonblank password, '.password' becomes a required field.

':d' is an optional drive specification. When omitted the system does an automatic search, starting with drive 0.

options

allows the user to specify runtime options. Each option must be specified as shown below, separated by spaces. The left and right parenthesis are required if any comments are present.

When no options are specified, the runtime will execute the User's program without Interactive Debug, with all switches set to 0, using all of available memory.

2.3.1 Runtime Options

D

'D' invokes the RSCOBOL Interactive Debug package. See RSCOBOL Interactive Debug discussion, below, for operating instructions.

The default is not to invoke Interactive Debug.

S=nn..n

'S' sets (or resets) the value of SWITCHES in the COBOL program.

Each 'n' is a switch value, 0 for off, 1 for on, numbered 1 to 8, left to right. Trailing zeroes need not be specified.

The default is to set all switches off (0).

T=hhhh

'T' sets the top of available memory to a value different from the highest available address. This is used to protect assembly language user subroutines, all of which must be created to load above the hexadecimal address 'hhhh'.

The default is to use all available memory.

2.3.2 Runtime Messages

Messages which report the runtime's status, or its ability to execute the COBOL program, are reported on the system display as they are detected.

TRS-80 Model I/III COBOL Runtime (RM/COBOL ver v.r)
Copyright 1980 by Tandy Corp. Licensed from Ryan-McFarland Corp.

Indicates that the runtime has been loaded and has begun to execute the specified program. 'ver v.r' identifies the version (v) and revision (r) level of the runtime.

COBOL STOP RUN AT xxyyyy IN nnnnnn

This is the normal termination message of a program.

'xxyyyy' identifies the overlay (xx) and statement address (yyyy) where the program terminated. 'nnnnnn' are the first six characters of the PROGRAM-ID.

If Debug was invoked on the command line, an 'S' Debug command may be used to cause Debug to exit to the operating system.

COBOL STOP literal AT xxyyyy IN nnnnnn
CONTINUE (Y/N)?

This message indicates that a STOP 'literal' statement has been encountered. 'xxyyyy' identifies the overlay (xx) and statement address (yyyy) where the program terminated. 'nnnnnn' are the first six characters of the PROGRAM-ID.

Responding with a 'Y' will be the equivalent of a "pause" statement, returning control to the next COBOL statement.

An 'N' response will cause all program files to be closed and control will be returned to the operating system.

2.3.3 Examples

RUNCOBOL PAYROLL (S=1011)

locates, loads, and executes the compiled COBOL program PAYROLL/COB; and sets the value of SWITCHES 1, 3, and 4 'on', all others 'off'.

RUNCOBOL MORTGAGE/TST:2 (D)

loads the compiled COBOL program MORTGAGE/TST from drive 2 along with the Interactive Debug package. Control is passed directly to Debug.

RUNTIME ERROR, NO: nnnn

an internal error has occurred which prevents continued execution. The value of 'nnnn' identifies the condition which caused the error.

0010 Unable to locate or load User Debug.
Install RSCBLDvr as described in the chapter on 'Installation Procedures'.

0020 Invalid TRSDOS

Execution was attempted under an incorrect version of TRSDOS Order correct version of TRSDOS.

Required TRSDOS versions are:

Model I - 1.1
Model III - 2.3

2.4 Runtime Diagnostics

Diagnostic messages are display if an internal error occurs, or if an I/O error occurs that was not, or could not, be processed by an appropriate USE procedure.

If Debug was invoked, Debug will be entered to allow examination of program data values; otherwise, control will return to the operating system.

COBOL error AT xxyyyy IN nnnnnn

Indicates an internal error condition has occurred, where 'error' identifies the error condition. 'xxyyyy' identifies the overlay (xx) and statement address (yyyy) where the program terminated. 'nnnnnn' are the first six characters of the PROGRAM-ID.

COBOL filename IO ERROR = cc AT xxyyyy IN nnnnnn

Identifies that an abnormal I/O condition, 'cc' has caused the program to be aborted. 'xxyyyy' identifies the overlay (xx) and statement address (yyyy) where the program terminated. 'nnnnnn' are the first 6 characters of the PROGRAM-ID.

The I/O error 'cc' has a different meaning depending on whether the file's organization is sequential, relative or indexed.

Sequential Files:

10 AT END.
The sequential READ statement was unsuccessfully executed as a result of an attempt to read a record when no next logical record exists in the file.

30 PERMANENT ERROR.
The input-output statement was unsuccessfully executed as the result of an input-output error, such as data check parity error, or transmission error. May also indicate attempted execution of an instruction not implemented in the runtime (REWRITE to a variable length record (VLR) file; CLOSE REEL). May also indicate that no more space is available on the disk.

34 PERMANENT ERROR BOUNDARY VIOLATION.
The input-output statement was unsuccessfully executed as the result of a boundary violation for a sequential file.

90 INVALID OPERATION.
An attempt has been made to execute a READ, WRITE, or REWRITE statement that conflicts with the current open mode or a REWRITE statement was not preceded by a successful READ statement.

91 FILE NOT OPENED.
An attempt has been made to execute a DELETE, READ, START, UNLOCK, WRITE, REWRITE or CLOSE statement on a file which is not currently open.

92 FILE NOT CLOSED.
An attempt has been made to execute an OPEN statement on a file which is currently open.

93 FILE NOT AVAILABLE.
An attempt has been made to execute an OPEN statement for a file closed with LOCK.

94 INVALID OPEN.
An attempt has been made to execute an OPEN statement for a file with no external correspondence or a file having inconsistent parameters.

95 INVALID DEVICE.
An attempt has been made to execute a CLOSE REEL statement, or to execute an OPEN statement for a file which is assigned to a device in conflict with the externally assigned device. Valid combinations are:

Program Assignment	External Assignment
RANDOM	Disk
INPUT	Disk
OUTPUT	Disk
PRINT	Disk, line printer
INPUT-OUTPUT	Disk

96 UNDEFINED CURRENT RECORD POINTER STATUS.
An attempt has been made to execute a READ statement after the occurrence of an unsuccessful READ statement without an intervening successful CLOSE and OPEN.

97 INVALID RECORD LENGTH.
An attempt has been made to execute a REWRITE statement when the new record length is different from that of the record to be rewritten, or to OPEN a file that was defined with a maximum record length different from the externally defined maximum record length, or to execute a WRITE statement that specifies a record with a length smaller than the minimum or larger than the maximum record size.

Relative and Indexed Files:

10 AT END.
The Format 1 READ statement was unsuccessfully executed as a result of an attempt to read a record when no next logical record exists in the file.

21 SEQUENCE ERROR FOR A SEQUENTIALLY ACCESSED INDEXED FILE.
The ascending sequence requirement of successive record key values has been violated or the record key value has been changed by the COBOL program between the successful execution of a READ statement and the execution of the next REWRITE statement for that file.

22 DUPLICATE KEY VALUE.
An attempt has been made to WRITE a record that would create a duplicate key on a file that does not allow duplicates.

23 NO RECORD FOUND.
An attempt has been made to access a record, identified by a key, and that record does not exist in the file.

24 BOUNDARY VIOLATION.
An attempt has been made to WRITE beyond the externally-defined boundaries of a file.

30 PERMANENT ERROR.
The input-output statement was unsuccessfully executed as the result of an input-output error, such as data check, parity error, or transmission error. May also indicate that no more space is available on the disk.

90 INVALID OPERATION.
An attempt has been made to execute a DELETE, READ, REWRITE, START, or WRITE statement which conflicts with the current open mode of the file or a sequential access DELETE or REWRITE statement not preceded by a successful read statement.

91 FILE NOT OPENED.
An attempt has been made to execute a CLOSE, DELETE, READ, REWRITE, START, UNLOCK, or WRITE statement on a file which is not in an open mode.

92 FILE NOT CLOSED.
An attempt has been made to execute an OPEN statement on a file that is currently open.

93 FILE NOT AVAILABLE.
An attempt has been made to execute an OPEN statement on a file closed with LOCK.

94 INVALID OPEN.
An attempt has been made to execute an OPEN statement for a file with no external correspondence or a file having inconsistent parameters.

95 INVALID DEVICE.
An attempt has been made to execute an OPEN statement on a file whose device description conflicts with the externally assigned device. The device must be RANDOM and the external correspondence must be a disk.

96 UNDEFINED CURRENT RECORD POINTER.
An attempt has been made to execute a Format 1 READ statement when the current record pointer has an undefined state. This can occur only as the result of a preceding unsuccessful READ or START statement.

97 INVALID RECORD LENGTH.
An attempt has been made to execute a REWRITE statement and the new record length is different from that of the record to be rewritten, or to OPEN a file that was defined with a maximum record length different from the externally defined maximum record length, or to execute a WRITE statement that specifies a record with a length smaller than the minimum or larger than the maximum record size.

98 INVALID INDEX.
An input-output statement on an indexed organization file was unsuccessful as a result of invalid data in the index. This can result if the externally assigned file is not an index organization file or if an undetected input-output error has occurred.

2.5 File System Considerations

Three types of files are supported by the COBOL Runtime: sequential, relative (random), and indexed sequential. These files exist on the disk as standard TRSDOS disk files. While the user will not typically need file information to execute COBOL programs, he is referred to the Technical Information Section of the Disk Operating System Reference Manual if further information is desired.

Files are specified in the user's program SELECT statement according to rules for the TRSDOS filespec, of the form:

```
filename/ext.password:d
```

where:

'filename' is required.

'/ext' is an optional name-extension.

'.password' is an optional password. Note: If the file was created with a nonblank password, '.password' becomes a required field.

':d' is an optional drive specification. When omitted the system does an automatic search, starting with drive 0.

2.5.1 COBOL Sequential Files

COBOL sequential files consist of a serially accessible set of 'logical' records. These 'logical' records can exist on the disk in either of two forms; 'variable' or 'fixed'.

'variable' records are identified in the File Description Entry (FD) by specifying "RECORD CONTAINS n TO m CHARACTERS". 'fixed' records are identified by specifying RECORD CONTAINS n CHARACTERS". The user is cautioned to maintain a consistent specification among all programs referring to the same file.

'variable' records contain a one byte length field at the beginning of each record, followed by the actual data bytes. The record length can vary from record to record. The second length byte indicates the entire length of the record, including the length byte. This can be any value from 2 to 255. This format is stored on the disk as one byte records (LRL=1).

'fixed' records are all of the same length and do not contain a length byte. These files exist on the disk as standard TRSDOS fixed length records of length (LRL=) 1 to 255 characters.

2.5.2 COBOL Relative Files

COBOL relative files are addressable randomly by 'logical' record number. These files exist on the disk as fixed length records.

COBOL relative file 'logical' records are internally formatted, and can be created and/or accessed only by COBOL programs. Each 'logical' record can have a maximum length of 253 bytes.

COBOL relative files are dynamically allocated or extended as required by TRSDOS.

2.5.3 COBOL Indexed Files

COBOL indexed files are created and maintained by the COBOL runtime; implemented on the disk using TRSDOS fixed length records of 256 bytes.

COBOL indexed files are internally formatted, and can be created and/or accessed only by COBOL programs. Each 'logical' record can have a maximum length of 4096 bytes.

Indexed files contain an index structure for each key specified interspersed with the data records. The use of ALTERNATE KEYS can cause a geometric increase in the time required to create the file; however, access time will be relatively constant throughout the file.

COBOL indexed files are dynamically allocated or extended as required by TRSDOS. The calculation below provides an approximation for the file space required for a given file:

```
NRECS = Int ((S + 33)/32) * R / 8

      + (R * 2) / Int (252/(Kn+8))        for each key

      + (R * D) / 8                       if duplicates
```

where:

R = maximum number of records desired
S = size of records (in bytes)
Kn = size of Kn (in bytes)
D = number of keys that allow duplicates

2.5.4 COBOL Label Processing

The COBOL language allows the specification of the existance, and processing, of Label records on file type devices.

TRSDOS provides automatic maintenance and validation of file specifications by name and file type. No additional Label processing is performed unique to COBOL programs or files.

References to Label processing in the file description entry (FD), OPEN statement, and CLOSE statement, are checked for correct syntax by the compiler. They are largely ignored by the runtime except that appropriate error codes will be returned, and any applicable USE procedures will be executed.

2.6 Runtime Memory Usage

The TRSDOS Operating System occupies lower memory from location 0000H to 05200H. The COBOL Runtime is loaded starting at 05200H. The remaining memory is allocated as follows:

The main COBOL object program is loaded immediately behind the COBOL Runtime. Space for COBOL overlays (SECTIONS greater than 50) are included in this area.

Additional COBOL programs are loaded behind this main program as they are CALLed (See the CALL statement below).

Assembly Language programs are loaded in high memory at the address they were assigned at 'DUMP' time (See Runtime 'T=hhhh' option).

File buffers are dynamically allocated from high memory downward, when OPENed, deallocated (space recovered for use by other files) when CLOSEd.

CHAPTER 3

INTERACTIVE DEBUG

3.1 Debug Overview

COBOL Interactive Debug is dynamically loaded when the user specifies the 'D' option on the RUNCOBOL statement. Debug is then given control and supervises the execution of the user's program.

Interactive Debug is loaded directly behind COBOL Runtime, requiring approximately 1000 bytes.

3.2 User Interaction and Display

All Debug commands, and all resultant displays, are through the system console.

Debug will request command input by a prompt of the form

```
nnnnnn xxyyyy
```

where 'nnnnnn' are the first 6 characters of PROGRAM-ID, 'xx' is the overlay number, and 'yyyy' is the hexadecimal location within the specified overlay that will be executed next.

The values of 'xx' and 'yyyy' are taken directly from the Debug column in the source listing for program 'nnnnnn'.

3.3 Debug Commands

All commands are specified by a single character, optionally followed by one or more arguments. Optional fields are shown surrounded by brackets; the brackets are never entered. All numeric arguments are in hexadecimal unless otherwise noted.

Invalid commands will be rejected with 'ERROR' displayed; corrected input will be requested with a reprompt.

A[xx]yyyy[,nnnnnn] Address stop.

Executes object instructions until overlay number 'xx' and location 'yyyy' in program nnnnnn is to be executed. Debug will regain control immediately prior to the execution of the specified COBOL sentence, and request further command input.

If 'xx' is specified, 'yyyy' must be fully four hexadecimal digits; if 'xx' is not specified, then leading zeros are not required for 'yyyy'. If 'nnnnnn' is omitted, it is assumed to be the first six characters of the program-id of the currently executing program.

S[n] Single step sentence.

Execute 'n' COBOL sentences and return to the debug monitor.

The decimal argument 'n' specifies the number of COBOL sentences to be executed before returning the Debug.

Dxxxx,yyyy[,tttt] Dump by type.

Display the COBOL data item starting at hexadecimal location 'xxxx' of decimal length 'yyyy' and type 'tttt'. The values for 'xxxx', 'yyyy', and 'tttt' are directly from the first three columns of the allocation map. 'tttt' may be one of the following:

NSU	NPS
NSS	ABS
NCU	ANS
NCS	GRP
NBS	ANSE
NSE	HEX (hexadecimal)

Dump Display has the format:

xxxx tttt dddd....

where dddd = data in the specified format

Note: Only items in the currently executing program can be displayed. This does not include linkage items.

Q Quit Execution.

Terminate Debug and force an immediate STOP RUN. Enter 'S' to return to TRSDOS.

E Exit

Exit the Debugger. Continue normal execution as if the debugger had not been invoked on the command line.

CHAPTER 4

SYSTEM CONSIDERATIONS

4.1 The ACCEPT and DISPLAY Statements

The ACCEPT and DISPLAY statements support the transfer of data between the keyboard and display and the User's program data area. These statements allow the specification of general phrases which may not be supported on every CRT.

Phrases which are not supported will compile correctly, but will be ignored at runtime, causing no operation to take place. The phrases which are not supported are:

ACCEPT....HIGH, LOW, BLINK.

DISPLAY....HIGH, LOW, BLINK, BEEP.

The ON EXCEPTION phrase of the ACCEPT statement is executed when an invalid character is entered. Invalid characters include the valid control characters (CNTR/n) below 020H, and non-ASCII characters above and including 080H.

When an invalid character is entered, its ASCI equivalent is placed in the specified data-name and the ON EXCEPTION phrase is executed. To determine which control character was entered, define the data-name as USAGE COMPUTATIONAL-1 and compare for its ASCII value.

Certain keys affect the operation of the ACCEPT statement, including:

<-	Erases the current character and moves the cursor back one position.
CLEAR	Backspace to the beginning of the field, erasing all characters in the field.

4.2 The CALL Statement

When 'CALLed' the first time, COBOL and Assembly Language programs are loaded by Runtime and entered at their initial location. These 'called' programs remain in memory as long as the 'calling' program is active; i.e., has not EXITed. Therefore, subsequent CALLs from the 'calling' program will enter the 'called' program directly, without requiring the 'called' program to be reloaded.

Once the 'calling' program has EXITed, all related 'called' programs are discarded and will be reloaded if subsequently CALLed by any program, including the previous 'calling' program. Regardless of the sequence of 'called' and 'calling' programs, all related files not explicitly closed are forced closed by the interface upon EXIT from a given 'called' program.

COBOL programs that are to be CALLed must have been previously compiled. The default filename-extension for a program name in a CALL statement is '/COB'. A compiled COBOL program will have the required extension. If the extension used is not '/COB', then it must be specified in the CALL statement.

Assembly language programs that are to be CALLed must be in TRSDOS LOAD command format as created by DUMP, with a filename extension other than '/COB'. Assembly language programs must reside in high memory, and the 'T=nnnn' option must be specified on the Runtime command line to protect all memory required by the routine. The user is responsible for ensuring that the assembler programs do not interfere with each other.

Assembly language programs are loaded and reused while the 'calling' program resides in memory. If the COBOL 'calling' program is reloaded in memory, then the assembler program will again be reloaded when it is called.

At entry time to an assembly-language routine register IX points to the parameter list defined by the USING clause of the CALL statement. The first word on the list contains the number of bytes in the list. Subsequent words are addresses of the USING arguments: e.g., if the length word specifies 6 bytes, there are 2 addresses following the length word. For example:

```
(IX) =>  DW  Argument List Length (n * 2 + 2)
         DW  USING Argument 1
         DW  USING Argument 2
         .
         .
         DW  USING Argument n
```

The format of each argument depends on its dataname PICTURE definition; see the COBOL Language Manual, 'the PICTURE Clause'.

At exit time from an assembler routine, register A may be set non-zero to request a STOP RUN.

4.3 The COPY Statement

The COPY statement provides the facility to copy (include) COBOL source text from a user-specified file into the source program. The complete file is copied into the program, without change, at the location of the COPY statement.

The file to be copied is identified in the COBOL program by the statement

```
COPY  filename
```

or

```
COPY  "filename/ext.password:d"
```

where:

'filename' is required.

'/ext' is an optional name-extension. When omitted the default '/CBL: is used.

'.password' is an optional password. Note: If the file was created with a nonblank password, '.password' becomes a required field.

':d' is an optional drive specification. When omitted the system does an automatic search, starting with drive 0.

A filename consisting only of letters and numbers (first character must be letter) can be written without surrounding quotes. All other forms must be surrounded by quotes.

Examples:

```
IDENTIFICATION DIVISION.
     COPY      STDID.
ENVIRONMENT DIVISION.
     COPY      "STDENVIR/TST".
DATA DIVISION.
     COPY      "STDDATA/CBL:1".
```

4.4 The WRITE...ADVANCING ZERO...Statement

The sequential WRITE statement allows control of the vertical positioning of each line on the printed page with the ADVANCING phrase.

The ... ADVANCING ZERO LINE(s) ... phrase allows overprinting on those print devices which support this feature. In all cases, the phrase will compile correctly, but may operate as though ...ADVANCING 1 LINE... was specified.

Standard Radio Shack Line Printers automatically advance after each line is printed. Therefore, the ...ADVANCING ZERO LINES... phrase will execute as ... ADVANCING 1 LINE. The Compiler and Runtime defaults to standard Radio Shack Line Printer operation.

CHAPTER 5

INSTALLATION PROCEDURES

Installation of RSCOBOL requires only that the object modules be copied from the Development and Runtime factory release diskettes to the appropriate user diskette. NOTE: 'nn' indicates the current release level, i.e., release 1.3 will be '13'.

The modules required to compile COBOL programs are:

RSCOBOL
RSCBL2nn/OBJ
RSCBL3nn/OBJ
RSCBL4nn/OBJ

The modules required to execute compiled COBOL programs are:

RUNCOBOL
RSCBLDnn/OBJ

As with all Development and Runtime factory release diskettes, the user should save it in a secure location in case re-creation is required.

APPENDIX A

SAMPLE PROGRAMS

```
TRS-80 Model I/III COBOL (RM/COBOL 1.3A)          10/31/80  00:15:44  PAGE     1
SOURCE FILE: CALCXMPL                        OPTION LIST: (P,T,O=2,X

LINE  DEBUG PG/LN  A...B.....................................................

   1                IDENTIFICATION DIVISION.
   2                PROGRAM-ID.
   3                    CALCULATOR.
   4
   5                ENVIRONMENT DIVISION.
   6                CONFIGURATION SECTION.
   7                SOURCE-COMPUTER.       RMC.
   8                OBJECT-COMPUTER.       RMC.
   9
  10                DATA DIVISION.
  11                WORKING-STORAGE SECTION.
  12                77  RESULT                 PICTURE S9(9)V9(9) VALUE ZERO.
  13                77  OPERAND-1              PICTURE S9(9)V9(9).
  14                77  OPERAND-2              PICTURE S9(9)V9(9).
  15                77  WAIT-CHAR              PICTURE  X.
  16                01  GREETING.
  17                    02  FILLER             PICTURE X(18)
  18                          VALUE "CALCULATOR PROGRAM".
  19                01  OPERATION-MESSAGE.
  20                    02  FILLER             PICTURE X(37)
  21                          VALUE "CHOOSE YOUR OPERATION (+,-,*,/) = ".
  22                01  OPERATOR               PICTURE X(2).
  23                01  RESULT-MESSAGE.
  24                    02  FILLER             PICTURE X(12)
  25                          VALUE "RESULT IS = ".
  26                    02  RESULT-EDITED      PICTURE -(9)9.9(9).
  27                    02  FILLER             PIC X(4) VALUE SPACES.
  28                    02  OVERFLOW-FIELD     PIC X(8) VALUE SPACES.
  29                01  WAIT-MESSAGE.
  30                    02  FILLER             PICTURE X(36)
  31                          VALUE "HIT NEWLINE TO CONTINUE (Q TO QUIT) ".
  32                01  OPERAND-1-MESSAGE.
  33                    02  FILLER             PICTURE X(12)
  34                          VALUE "OPERAND-1 = ".
  35                01  OPERAND-2-MESSAGE.
  36                    02  FILLER             PICTURE X(12)
  37                          VALUE "OPERAND-2 = ".
```

SOURCE FILE: CALCXMPL OPTION LIST: (P,T,O=2,X

```
LINE  DEBUG PG/LN  A...B.....................................................

  38                /    EJECT
  39                 PROCEDURE DIVISION.
  40   >0000        RESIDENT SECTION 1.
  41   >0000        NOT-START.
  42   >0000            GO TO DISPLAY-GREETING.
  43   >0004        RE-TRY.
  44   >0004            DISPLAY OPERATION-MESSAGE, LINE 2, ERASE.
  45   >000C            ACCEPT OPERATOR, POSITION 0, PROMPT, ECHO.
  46   >0014            IF OPERATOR EQUAL  "+ " GO TO ADDITION.
  47   >001C            IF OPERATOR EQUAL  "- " GO TO SUBTRACTION.
  48   >0024            IF OPERATOR EQUAL  "* " GO TO MULTIPLICATION.
  49   >002C            IF OPERATOR EQUAL  "/ " GO TO DIVI-SION.
  50   >0034            IF OPERATOR EQUAL  "Q " GO TO END-RUN.
  51   >003C            GO TO RE-TRY.
  52   >003E        DISPLAY-RESULT.
  53   >003E            MOVE RESULT TO RESULT-EDITED.
  54   >0042            DISPLAY RESULT-MESSAGE.
  55   >0046            MOVE ZERO TO RESULT.
  56   >004A            MOVE SPACES TO OVERFLOW-FIELD.
  57   >0050        WAIT-ENTRY.
  58   >0050            DISPLAY WAIT-MESSAGE.
  59   >0054            ACCEPT WAIT-CHAR, POSITION 0, PROMPT, ECHO.
  60   >005C            IF WAIT-CHAR EQUAL  "Q" GO TO END-RUN.
  61   >0064            GO TO RE-TRY.
  62   >0066        GET-OPERANDS.
  63   >0066            DISPLAY OPERAND-1-MESSAGE, LINE 4.
  64   >006C            ACCEPT OPERAND-1, LINE 4, POSITION 13, SIZE 10,
  65                         PROMPT, CONVERT.
  66   >0078            MOVE OPERAND-1 TO RESULT-EDITED.
  67   >007C            DISPLAY RESULT-EDITED, LINE 4, POSITION 13.
  68   >0084            DISPLAY OPERAND-2-MESSAGE.
  69   >0088            ACCEPT OPERAND-2 , LINE 5, POSITION 13, SIZE 10,
  70                         PROMPT, CONVERT.
  71   >0094            MOVE OPERAND-2 TO RESULT-EDITED.
  72   >0098            DISPLAY RESULT-EDITED, LINE 5, POSITION 13.
  73   >00A2        END-RUN.
  74   >00A2            EXIT PROGRAM.
  75   >00A6        STOP-RUN.
  76   >00A6            STOP RUN.
```

```
LINE  DEBUG PG/LN  A...B.....................................................

  77               /    EJECT
  78>0100A8         OVERLAY-ADDITION SECTION 51.
  79>0100A8         ADDITION.
  80>0100A8             PERFORM GET-OPERANDS.
  81>0100AA             ADD OPERAND-1    OPERAND-2 GIVING RESULT
  82                        ON SIZE ERROR MOVE "OVERFLOW" TO OVERFLOW-FIELD.
  83>0100B8             GO TO DISPLAY-RESULT.
  84
  85>0200A8         OVERLAY-SUBTRACTION SECTION 52.
  86>0200A8         SUBTRACTION.
  87>0200A8             PERFORM GET-OPERANDS.
  88>0200AA             SUBTRACT OPERAND-2 FROM OPERAND-1 GIVING RESULT
  89                        ON SIZE ERROR MOVE "OVERFLOW" TO OVERFLOW-FIELD.
  90>0200B8             GO TO DISPLAY-RESULT.
  91
  92>0300A8         OVERLAY-MULTIPLICATION SECTION 53.
  93>0300A8         MULTIPLICATION.
  94>0300A8             PERFORM GET-OPERANDS.
  95>0300AA             MULTIPLY OPERAND-1 BY OPERAND-2 GIVING RESULT
  96                        ON SIZE ERROR MOVE "OVERFLOW" TO OVERFLOW-FIELD.
  97>0300B8             GO TO DISPLAY-RESULT.
  98
  99>0400A8         OVERLAY-DIVISION SECTION 54.
 100>0400A8         DIVI-SION.
 101>0400A8             PERFORM GET-OPERANDS.
 102>0400AA             DIVIDE OPERAND-1 BY OPERAND-2 GIVING RESULT ROUNDED
 103                        ON SIZE ERROR MOVE "OVERFLOW" TO OVERFLOW-FIELD.
 104>0400BA             GO TO DISPLAY-RESULT.
 105
 106>0500A8         OVERLAY-DISPLAY-GREETING SECTION 98.
 107>0500A8         DISPLAY-GREETING.
 108>0500A8             DISPLAY GREETING.
 109>0500AC             GO TO WAIT-ENTRY.
 110
 111               END PROGRAM.
```

SOURCE FILE: CALCXMPL OPTION LIST: (P,T,O=2,X

ADDRESS	SIZE	DEBUG	ORDER	TYPE	NAME
>0004	19	NSS	0	NUMERIC SIGNED	RESULT
>0018	19	NSS	0	NUMERIC SIGNED	OPERAND-1
>002C	19	NSS	0	NUMERIC SIGNED	OPERAND-2
>0040	1	ANS	0	ALPHANUMERIC	WAIT-CHAR
>0042	18	GRP	0	GROUP	GREETING
>0054	37	GRP	0	GROUP	OPERATION-MESSAGE
>007A	2	ANS	0	ALPHANUMERIC	OPERATOR
>007C	44	GRP	0	GROUP	RESULT-MESSAGE
>0088	20	NSE	0	NUMERIC EDITED	RESULT-EDITED
>00A0	8	ANS	0	ALPHANUMERIC	OVERFLOW-FIELD
>00A8	36	GRP	0	GROUP	WAIT-MESSAGE
>00CC	12	GRP	0	GROUP	OPERAND-1-MESSAGE
>00D8	12	GRP	0	GROUP	OPERAND-2-MESSAGE

READ ONLY BYTE SIZE = >01BE

READ/WRITE BYTE SIZE = >00EC

OVERLAY SEGMENT BYTE SIZE = >002E

TOTAL BYTE SIZE = >02D8

0 ERRORS

0 WARNINGS

CROSS REFERENCE /DECL/ *DEST*

```
ADDITION                        0046   /0079/
DISPLAY-GREETING                0042   /0107/
DISPLAY-RESULT                 /0052/  0083   0090   0097   0104
DIVI-SION                       0049   /0100/
END-RUN                         0050   0060   /0073/
GET-OPERANDS                   /0062/  0080   0087   0094   0101
GREETING                       /0016/  0108
MULTIPLICATION                  0048   /0093/
NOT-START                      /0041/
OPERAND-1                      /0013/ *0064*  0066   0081  *0088*  0095   0102
OPERAND-1-MESSAGE              /0032/  0063
OPERAND-2                      /0014/ *0069*  0071   0081   0088  *0095*  0102
OPERAND-2-MESSAGE              /0035/  0068
OPERATION-MESSAGE              /0019/  0044
OPERATOR                       /0022/ *0045*  0046   0047   0048   0049   0050
OVERFLOW-FIELD                 /0028/ *0056* *0082* *0089* *0096* *0103*
OVERLAY-ADDITION               /0078/
OVERLAY-DISPLAY-GREETING       /0106/
OVERLAY-DIVISION               /0099/
OVERLAY-MULTIPLICATION         /0092/
OVERLAY-SUBTRACTION            /0085/
RESIDENT                       /0040/
RESULT                         /0012/  0053  *0055* *0081* *0088* *0095* *0102*
RESULT-EDITED                  /0026/ *0053* *0066*  0067  *0071*  0072
RESULT-MESSAGE                 /0023/  0054
RE-TRY                         /0043/  0051   0061
STOP-RUN                       /0075/
SUBTRACTION                     0047   /0086/
WAIT-CHAR                      /0015/ *0059*  0060
WAIT-ENTRY                     /0057/  0109
WAIT-MESSAGE                   /0029/  0058
```

```
TRS-80 Model I/III COBOL (RM/COBOL 1.3A)          10/31/80  00:22:43  PAGE     1
SOURCE FILE: ERRXMPL                        OPTION LIST: (T,P,O=2,X

L    DEBUG PG/LN  A...B.....................................................

   1          000010 IDENTIFICATION DIVISION.
   2          000020
   3          000030 PROGRAM-ID.
   4          000040     ERROR-EXAMPLES.
   5          000050
   6          000060 ENVIRONMENT DIVISION.
   7          000070 CONFIGURATION SECTION.
   8          000080 SOURCE-COMPUTER.   RMC-MINI.
   9          000090 OBJECT-COMPUTER.   RMC-MINI.
  10          000100 INPUT-OUTPUT SECTION.
  11          000110 FILE-CONTROL.
  12          000120     SELECT INPUT-FILE
  13          000130         ASSIGN TO INPUT, INPUT-NAME;
  14          000140         FILE STATUS IS INPUT-STATUS.
  15          000150     SELECT OUTPUT-FILE
  16          000160         ASSIGN TO OUTPUT, OUTPUT-NAME;
  17          000170         FILE STATUS IS OUTPUT-STATUS.
  18          000180
  19          000190 DATA DIVISION.
  20          000200 FILE SECTION.
  21          000210 FD  INPUT-FILE
  22          000220     RECORD CONTAINS 80 CHARACTERS
  23          000230     LABEL RECORD IS OMITTED.
  24          000240 01  INPUT-REC.
  25          000250     05  FILLER           PIC  X(06).
  26          000260     05  INPUT-FLD        PIC  X(66).
  27          000270     05  AREA-FLDS        REDEFINES INPUT-FLD.
  28          000280         10  AREA-C       PIC  X(01).
  29          000290         10  AREA-A       PIC  X(04).
  30          000300         10  AREA-B       PIC  X(61).
  31          000310     05  FILLER           PIC  X(08).
  32          000320 FD  OUTPUT-FILE
  33          000330     RECORD CONTAINS 80 CHARACTERS;
  34          000340     LABEL RECORD IS OMITTED.
  35          000350 01  OUTPUT-REC.
  36          000360     05  SEQ-FLD          PIC  9(06).
  37          000370     05  OUTPUT-FLD       PIC  X(66).
  38          000380     05  FILLER           PIC  X(08).
  39          000390 WORKING-STORAGE SECTION.
  40          000400 77  INPUT-NAME           PIC  X(28).
  41          000410 77  OUTPUT-NAME          PIC  X(28).
  42          000420 77  COUNT                PIC  9(06) VALUE 0.
  43          000430 77  LARGE-VALUE          PIC  X(04) VALUE "ERROR".
  44          000440 77  PIC-ERROR            PIC  *(05).*9.
                                                                        $
*****    1) PICTURE  *E*E*E*
```

```
TRS-80 Model I/III COBOL (RM/COBOL 1.3A)          10/31/80  00:22:43  PAGE     2
SOURCE FILE: ERRXMPL                        OPTION LIST: (T,P,O=2,X

LINE  DEBUG PG/LN  A...B.....................................................

  49                /
  50          000490 PROCEDURE DIVISION.
  51    >0000 000500 0100.
  52    >0000 000510      DISPLAY "COBOL PROGRAM SEQUENCER",
  53          000520          LINE 1 POSITION 30 ERASE.
  54    >000A 000530      DISPLAY SPACES LINE 2.
  55    >0010 000540      DISPLAY "INPUT FILE:  ".
  56    >0014 000550      MOVE 3.5 TO OUTPUT-STATUS.
                                    $              $
*****    1) MOVE    *E*E*E*
```

```
ADDRESS  SIZE DEBUG ORDER TYPE                    NAME

            0              FILE                   INPUT-FILE
 >0000     80  GRP     0   GROUP                   INPUT-REC
 >0006     66  ANS     0   ALPHANUMERIC              INPUT-FLD
 >0006     66  GRP     0   GROUP                     AREA-FLDS
 >0006      1  ANS     0   ALPHANUMERIC                AREA-C
 >0007      4  ANS     0   ALPHANUMERIC                AREA-A
 >000B     61  ANS     0   ALPHANUMERIC                AREA-B

            0              FILE                   OUTPUT-FILE
 >0050     80  GRP     0   GROUP                   OUTPUT-REC
 >0050      6  NSU     0   NUMERIC UNSIGNED          SEQ-FLD
 >0056     66  ANS     0   ALPHANUMERIC              OUTPUT-FLD

 >00A4     28  ANS     0   ALPHANUMERIC           INPUT-NAME

 >00C0     28  ANS     0   ALPHANUMERIC           OUTPUT-NAME

 >00DC      6  NSU     0   NUMERIC UNSIGNED       COUNT

 >00E2      4  ANS     0   ALPHANUMERIC           LARGE-VALUE

 >00E6    166  ANS     0   ALPHANUMERIC           PIC-ERROR

 >00EE      4  ANS     0   ALPHANUMERIC           INPUT-STATUS

 >00F2      2  ANS     0   ALPHANUMERIC           OUTPUT-STATUS

 >00F4      6  NSU     0   NUMERIC UNSIGNED       SEQ-VALUE

RESERVED WORD CONFLICT *W*W*W*W*W*W*W*W*W*W*W*W*W  COUNT

ILLEGAL PERFORM *E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E INPUT-CHECK

UNDEFINED PROCEDURE *E*E*E*E*E*E*E*E*E*E*E*E*E*E*E 0150

VALUE ERROR *E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E LARGE-VALUE

FILE STATUS ERROR *E*E*E*E*E*E*E*E*E*E*E*E*E*E*E*E INPUT-FILE

READ ONLY BYTE SIZE =          >017E

READ/WRITE BYTE SIZE =         >0138

OVERLAY SEGMENT BYTE SIZE = >0000

TOTAL BYTE SIZE =              >02B6

11 ERRORS

 8 WARNINGS
```

```
CROSS REFERENCE                  /DECL/ *DEST*

AREA-A                           /0029/
AREA-B                           /0030/
AREA-C                           /0028/
AREA-FLDS                        /0027/
COUNT                            /0042/ *0073*  0076
INPUT-CHECK                       0068
INPUT-FILE                       /0012/ /0021/
INPUT-FLD                        /0026/ /0027/  0071
INPUT-NAME                       *0013* /0040/ *0057*
INPUT-REC                        /0024/
INPUT-STATUS                     *0014* /0045/
LARGE-VALUE                      /0043/
OUTPUT-FILE                      /0015/ /0032/  0061
OUTPUT-FLD                       /0037/ *0071*
OUTPUT-NAME                      *0016* /0041/ *0059*
OUTPUT-REC                       /0035/ *0062* *0072*
OUTPUT-STATUS                    *0017* /0046/
PIC-ERROR                        /0044/
SEQ-FLD                          /0036/ *0070*
SEQ-VALUE                        /0047/ *0063* *0069*  0070
0100                             /0051/
0150                              0080
0200                             /0065/  0074
0300                              0067  /0075/
```

APPENDIX B

TRS-80 (TM) MODEL I/III COBOL

SAMPLE SESSION

This section will take you through a compilation and execution session, starting with a COBOL source file. We will use the sample program, CALCXMPL/CBL, included with your COBOL diskettes.

Note for Model III users: References will be made to the separate Development and Runtime diskettes. Since Model III diskettes will hold the complete system, your one diskette will take the place of both the Development and Runtime diskettes.

STEP ONE. Create the source file.

In this session, we will use the sample program, CALCXMPL/CBL, for the source file. To create your own source file, follow the instructions in the COBOL Editor (CEDIT) User's Guide.

STEP TWO. Compile.

When compiling (RSCOBOL), the COBOL Development diskette must be in one of the drives. The program being compiled must also be on a diskette, although not necessarily on the same one as RSCOBOL. Our sample program is on both the Development and the Runtime diskettes. Also, there must be some free space on one of the diskettes for the Compiler to write the compiled version of your program.

With the COBOL Development diskette in one of the drives, type under TRSDOS READY:

```
RSCOBOL CALCXMPL (T)
```

The T option causes a listing to be displayed at the console. See Section 1.3.2 in the COBOL USER'S GUIDE for other Compiler options that are available.

This command creates an object file that can be executed by the COBOL Runtime. This file will automatically be named CALCXMPL/COB. Compiled programs are always written to disk with the /COB extension and will be written on the first available diskette that has enough free space.

STEP THREE. Execute.

Model I users take out the Development diskette and replace it with the Runtime diskette. Also be sure that the diskette

containing the newly compiled version of our sample program is still on one of the drives.

Under TRSDOS READY, type:

```
RUNCOBOL CALCXMPL
```

The Runtime will execute the program CALCXMPL/COB. See Section 2.3.1 of the COBOL User's Guide for Runtime options.

C O N V E R S I O N S E S S I O N
F O R M O D E L I I I U S E R S

The diskettes you have contain all the files needed to compile and run COBOL programs. However, these diskettes are formatted for a Model I and need to be converted to Model IIIbefore you can use them. You will need one blank formatted diskette for this procedure.

First, BACKUP your Model III system diskette to the blank diskette. Take out your old Model III system disk and move the newly created system disk to drive 0. Use the PURGE:0 (SYS) command to delete all user files and all unnecessary system files. CONVERT/CMD is the only system file that is essential for the following procedure. You must have at least 130 free granules of space on the new system diskette. Check the directory to see how much space you do have.

Place the COBOL Development diskette in drive 1. Then use the conversion utility as shown below.

```
TRSDOS Ready
CONVERT  <ENTER>
```

The conversion utility will return with a prompt for Source Drive (you will enter 1) and Destination Drive (you will enter 0). The password on both the Development and the Runtime diskette is 'PASSWORD'.

The utility will convert the files to Model III format, writing the converted version onto the diskette in drive 0. Some of the files are passworded and the utility will prompt you as in the example shown below:

```
Enter Password for RSCOBOL/CMD ?
```

Just press <ENTER> and the files will be converted and transferred. Passwording does not prevent you in any way from using them.

Five of the files are passworded and you will have to press <ENTER> after every prompt. Four files are not passworded and will automatically be converted and written on drive 0.

When the conversion is complete the utility displays a

message telling you that it is done and then returns control to TRSDOS.

Put the COBOL Runtime diskette in drive 1 and once again use the CONVERT utility the same way as described above. There are some passworded files on this diskette also, so you will have to press <ENTER> when asked for the file passwords. Also, some of the files are stored on both diskettes. When trying to CONVERT the file the second time you will get the following message:

CALCXMPL/COB Existing file. Use it (Y/N/Q)?

Type N to use the previously converted file. The Y option will Convert the file again unnecessarily and the Q option will stop the CONVERT utility. To have more free space on the diskette you may PURGE the CONVERT utility when the conversion is complete, but it is not necessary. Label this new diskette to show that it contains the complete COBOL package.

We suggest that you make backups of your new COBOL diskette or keep the Model I version COBOL diskettes. This will give you some security against losing your COBOL package.

You may want a diskette with just the minimal Runtime files on it for running previously compiled programs. You will need a blank formatted diskette. BACKUP your COBOL diskette onto this new diskette. Then use the PURGE command to delete all but the necessary Runtime files. The only files that you need to keep on the new diskette are RUNCOBOL/CMD and RSCBLDnn/OBJ. (nn refers to the version number.)

Remember that only programs that have been already compiled using RSCOBOL can be used with this Runtime diskette.

TRS-80™ MODEL I/III

RSCOBOL CEDIT USER'S GUIDE

Using CEDIT to Create and Edit COBOL Source Files.

Radio Shack TRS-80 SOFTWARE™

CUSTOM MANUFACTURED IN THE USA FOR RADIO SHACK A DIVISION OF TANDY CORP.

TRS-80™

TRS-80 (TM) MODEL I/III COBOL

C E D I T
SOURCE PROGRAM EDITOR

USER'S GUIDE

Radio Shack®

TABLE OF CONTENTS

INTRODUCTION

CEDIT lets you create and edit COBOL source files (the files that are input to the COBOL Compiler).

Capabilities and features:

- Allows you to load in ("chain") multiple source files.
- Single-key abbreviations for many commands
- Powerful intra-line editing mode
- "M" command informs you of memory used/free at any time
- Global string find/change commands
- Editor provides line numbers in the range 0-65535

SOURCE FILE FORMAT

Source files are written to disk in the format required by the COBOL compiler, as follows:

1. Files are fixed-length record (FLR) type, LRL=256, as described in the TRSDOS Reference Manual.

2. Each record in the file corresponds to one line of source program. The first six data bytes in a record represent the sequence number in ASCII form followed by the COBOL source code. The carriage return (<ENTER>) used to terminate the line during line insertion is stored.

3. Text is stored exactly as it is displayed on the video, e.g., spaces are stored as spaces, not as a tab character.

TO START THE EDITOR

The editor program is included on the COBOL program diskette. It has the file name CEDIT.

To use the editor, put the COBOL diskette into one of your drives, and under TRSDOS READY, type:

```
CEDIT
```

The editor will start up with the prompt:

```
TRS-80 Cobol Editor  Ver v.r
Copyright (c) 1980 Tandy Corp.

>
```

Where v is the version and r is the release number. The > indicates you are in the command mode.

MODES OF OPERATION

There are three modes of operation:
. COMMAND, for entering the editor commands
. INSERT, for entering your text lines
. EDIT, for interactive editing of a line of text

COMMAND MODE

The > prompt followed by the blinking cursor indicates the editor is waiting for you to type in a command. Every command must be completed by pressing <ENTER>. To cancel a command, press <BREAK>.

INSERT MODE

You enter text one line at a time; a line consists of up to 255 characters, including the five-digit line number provided by CEDIT. Line numbers can range from 0 to 65535.

The I command puts you in the insert mode. When you start inserting a line, the editor displays the five-digit line number followed by the blinking cursor. Your text can begin in column seven. (See the COBOL Language Reference Manual for column-field uses in COBOL source programs.)

To store the current line, press <ENTER>. The editor will display the next line number, and you can begin inserting into that line. To cancel the current line and return to the command mode, press <BREAK>. See I Command for details.

EDIT MODE

There are many powerful edit sub-commands--identical in most cases to those in Model I and III BASIC's Edit Mode. There is also a sub-edit insertion mode in which the keys you type are inserted into the line at the current cursor position.

To start editing a line, use the E command. After editing the line, press <ENTER> to save the corrected line and return to the command mode. To cancel all changes made and return to the command mode, press <Q>. For further details, see E Command.

USING THE COMMAND MODE

Special terms used in the command descriptions:

"text", "text buffer", "text area"
All refer to the COBOL source program currently in RAM.

"current line"
The line most recently inserted, displayed or referenced in a command. When there is no text in RAM, current line is set to 100. Immediately after a file is loaded, the current line is set to the beginning of the text.

"increment"
The value which is added to the current line number whenever the editor needs to compute a new line number. After startup, loading a new file, and when there is no text in RAM, the increment is set to 10.

"line-reference"
Either an actual line number from 0 to 65535, or one of the following special abbreviations:

Symbol	Meaning
#	Beginning line of text (lowest-numbered line)
.	Current line
*	Last line of text (highest-numbered line)

"line-range"
This can be either a single-line reference or a pair of line-references separated by a colon:

Sample Command	Meaning
P100	Prints line 100 only
P100:300	Prints all lines from 100 to 300
P#:.	Prints all lines from beginning to current

"delimiter"
A special character used to delimit (mark the beginning and end of) a string. Any of the following characters can be used:

! " # $ % & ' ( ) * + , - . / : ; < = > ?

Whichever character is used to mark the beginning of a string must also be used to mark the end of the string.

Sample use...	Marks this string...
'THIS " MARK'	THIS " MARK
/X'8000'/	X'8000'
&~~~~~~~&	~~~~~~~ (seven blanks)

(The "~" symbol represents a blank space. It is used only where necessary for emphasis or illustration.)

SPECIAL KEYS IN THE COMMAND MODE

<BREAK>
Press this key to cancel the command you are entering, or to abort a command which is currently being executed.

<right-arrow>
Advances the cursor to the next four-column boundary (boundaries are at columns 4, 8, 12, ...)

<ENTER>
Pressing this key at the beginning of a command line displays the current line.

<up-arrow>
Pressing this key at the beginning of a command line displays the line which precedes the current line.

<down-arrow>
Pressing this key at the beginning of a command line displays the next line after the current line.

shift<left-arrow>
Erases the command you are entering.

<@>
Pauses H and P commands. Press any other key to continue.

COMMANDS

Note: Spaces are not significant in command lines. For example,
P 1 : 5
has the same effect as
P1:5
The P command is explained later on.

B

Displays the bottom line (last line in the text area).

C/search-string/replacement-string/n

Finds, changes, and displays the first n lines, from the current line, that contain search-string. In each of these lines search-string is changed to replacement-string. ONLY THE FIRST OCCURRENCE OF search-string IN A SINGLE LINE IS COUNTED AND CHANGED. If the end of text is reached before n finds, the message "string not found" will be displayed.

Upon completion of the command, the current line is set to the line of the last find, or to the first line of text when "string not found" is displayed.

/search-string/ is a sequence of characters delimited by a matched pair of characters from the set:

! " # $ % & ' ( ) * + , - . / : ; < = > ?

replacement-string/ is a sequence of characters terminated by the same character used to delimit search-string.

n Tells the maximum number of "changes" you want. n can be a number or an asterisk. The asterisk means change and list all occurrences. If n is omitted, only the first occurrence is changed and listed.

Sample Commands	Notes
C/VAR=/NET=/	Changes the first occurrence of "VAR=" to "NET=" in the first line that contains it.
C"VAR="NET="	Same as above.

C/RETRY/R/4	Changes the first occurrence of "RETRY" to "R" in the first four lines that contain it.
C/MISPELING/MIS-SPELLING/*	Changes the first occurrence of "MISPELING" to "MIS-SPELLING" in every line that contains it.
C/EXTRA//*	Changes the first occurrence of "EXTRA" to "" (null string) i.e., deletes the first "EXTRA" in every line that contains it.

D line-range

Deletes lines in the specified range and renumbers the following lines using the current increment. If line-range is omitted, the current line is deleted.

Sample Commands	Notes
D. or D	Deletes the current line.
D2	Deletes line number 2.
D98:115	Deletes lines found in the range 98 to 115.
D1000:*	Deletes all lines numbered 1000 or higher to end of text.

E line-reference

Starts edit mode using the specified line. If line-reference is omitted, the current line is used.

Edit sub-commands:

<ENTER>	Ends editing and returns to command mode.
shift<up-arrow>	Causes escape from sub-edit insertion (X, I, and H sub-commands) and returns to edit mode.
n <SPCBAR>	Advances cursor n columns. If n is omitted, 1 is used.
L	"Lists" working copy of the line and starts a new working copy.
X	"Extends" line: positions cursor to end of line and enters sub-edit insertion mode. Use shift<up-arrow> to escape to edit mode.
I	Enters sub-edit "insertion" mode at the

	current cursor position; use shift<up-arrow> to escape to edit mode.
A	("Again") Cancels changes and starts a new working copy of the line.
E	("End") Saves edited line and exits to command mode, > prompt.
Q	("Quit") Cancels changes and returns to command mode, > prompt.
H	"Hacks" remainder of line beginning at current cursor position and enters sub-edit insertion mode. Use shift<up-arrow> to escape to edit mode.
nD	"Deletes" n characters beginning at current cursor position. If n is omitted, 1 is used. The deletion is not echoed; use <L> to see the line with characters deleted.
nC	"Changes" next n characters from the current cursor position, using the next n characters typed. If n is omitted, 1 is used.
nSc	("Search") Moves cursor to nth occurrence of character c. Search starts at next character after the cursor. If n is omitted, 1 is used.
nKc	("Kill") Deletes all characters from current cursor position up to nth occurrence of character c, counting from current cursor position. If n is omitted, 1 is used. The deletion is not echoed; use <L> to see the line with characters deleted.

F/search-string/n

Finds and displays the first n lines which contain search-string, starting at the current line. ONLY THE FIRST OCCURRENCE OF search-string IN A SINGLE LINE IS COUNTED. If the end of text is reached before n finds, the message "string not found" will be displayed.

Upon completion of the command, the current line is set to the line of the last find, or to the first line of text when "string not found" is displayed.
/search-string/ is a sequence of characters delimited by

a matched pair of delimiters chosen from the set:

! " # $ % & ' ( ) * + , - . / : ; < = > ?

n Tells the maximum number of "finds" you want. n can be a number or an asterisk. The asterisk means find and list all occurrences. If n is omitted, only the first occurrence is listed.

Sample Commands	Notes
F/VAR=/	Finds and displays the first line that contains the string "VAR=".
F"VAR="	Same as above.
F/RETRY/4	Finds and displays the first eight lines containing at least one occurrence of "RETRY".
F/MISPELING/*	Finds and displays every line containing at least one occurrence of "MISPELING".

H line-range

("Hard-copy") Lists to the printer all lines found in the specified range. If line-range is omitted, all the lines after and including the current line are printed.

The printer should be initialized (with FORMS) before you execute this command.

Sample Commands	Notes
H#:*	Lists all lines to the printer.
H7020	Lists line 7020 to the printer.
H672:800	Lists all lines found in the range 672 to 800.

I start-line, increment

Starts the insert mode.

start-line is a line-reference telling the editor where to begin inserting into the text. If omitted, the current line is used.

,increment is a number telling the editor how to compute successive line numbers. If omitted, the current increment is used.

next line number (start-line + increment).

Special Keys in the Insert Mode

->	Advances the cursor to the next eight-column boundary (8, 16, 24, ...).
shift <-	Erases the line and starts over.
<-	Backspaces the cursor and erases the character.
<ENTER>	Marks the end of the current line. The editor will store the current line and start a new one, using increment to generate the next line number.

Overwriting lines
An automatic line numberer is provided to prevent you from accidentally overwriting lines already entered. If a line number conflict occurs the complete file will be renumbered from the current start-line by the current increment.

Sample Commands	Notes
I	Start inserting at current line number, using current increment.
I,1	Start inserting at current line number, using 1 as an increment. If current line number is in use, start with current line plus 1.
I45,2	Start inserting at line 45 with an increment of 2. If line 45 is in use, start with line 47.
I100	Start inserting at line 100, using the current increment. If line 100 is in use, start with 100 plus increment.

L filespec

Loads a source file from disk. If there is already text in RAM, the editor will ask whether you want to chain the new text onto the end of the old, or clear out the old first. If you chain the new text onto the old, the line numbers will start at the current start-line and be incremented by the current increment.

filespec is a TRSDOS file specification for a FLR text file. The file may have been created by this COBOL editor or by another means. However, it must be in the COBOL source file format. (See Source File Format.)

Sample Commands	Notes
L DEMO/BAS:1	Load DEMO/BAS from drive 1.
L XDATA	Load XDATA

M

Prints the number of characters in the source text (excluding the editor's line numbers) and the amount of memory free for text storage.

Sample Command	Notes
M	A typical response in a 48K system might might look like this: 000427- TEXT 039383- MEMORY Meaning you have 427 bytes of text, and 39383 free bytes of memory available.

N start-line,increment

Renumbers the entire text.

Note: The renumbering commands DO NOT RENUMBER LINE REFERENCES inside your program text; do not use them unless you are not concerned wth line references (GOTO, IF...THEN ..., GOSUB, etc.). To renumber your program properly, use the Compiler COBOL RENUMBER command.

start-line becomes the lowest line number when the text is renumbered. If start-line is omitted, the current line number is used.

increment is used in computing successive line numbers. If omitted, the current increment is used.

The current line before numbering is also the current line after renumbering.

Sample Commands	Notes
N	Repeats the last renumbering command.
N100	Renumbered text will start with line 100; successive lines computed with the

	current value of increment.
N100,25	As above; line numbers at increments of 25.

P line-range

Prints the specified lines to the display. If line-range is omitted, 14 lines starting at the current line are displayed.

Sample Commands	Notes
P	Prints 14 lines starting at current line.
P233	Prints line 233.
P.	Prints the current line.
P*	Prints the last line.
P140:615	Prints the lines within the specified range. Lines 140 and 615 don't have to be existing line numbers.

Q

Terminates session and returns to TRSDOS. The source text is not written to disk.

R line-reference, increment

Replaces contents of the specified line and continue in insert mode. If line-reference is omitted, the current line is used. If increment is omitted, the current increment is used. Also renumbers the complete file using the current start-line and the new increment.

The R command is equivalent to the D (delete) command followed by the I (insert) command. When you enter the command, the editor deletes the specified line and puts you into the insert mode, starting with the line just deleted.
After you press <ENTER>, the editor will contine in the insert mode, prompting you to enter the text of the next line number.
To escape from the insert mode, press <BREAK>.

Sample Commands	Notes
R125,3	Prompts you to insert replacement text for line 125. Subsequent line numbers will be generated with an increment of 3.

R*	Prompts you to insert replacement text for the highest numbered line in the text area; subsequent lines will be generated using the current increment.

T

Displays the top line (first line in the text area)

W filespec

Writes the text in RAM into the specified file.

filespec is a TRSDOS file specification. If file already exists, its previous contents will be lost.

Sample Commands	Notes
W DEMO/CBL:1	Save DEMO/CBL onto drive 1.
W XDATA	Save XDATA/CBL onto first available drive.

X/search-string/replacement-string/n

This command is exactly like the C (Change) command, except that it displays the line to be changed and queries you (Change?) each time it finds search-string. If you answer Y, the line will be changed; any other answer leaves the line unchanged. In either case, the process continues until all first occurrences have been found.

Sample Command	Notes
X/MISPELING/MSP/*	Changes the first occurrence of "MISPELING" to "MSP" in every line that contains it, but asks you to confirm each change before it is made.

RADIO SHACK, A DIVISION OF TANDY CORPORATION

U.S.A.: FORT WORTH, TEXAS 76102
CANADA: BARRIE, ONTARIO L4M 4W5

TANDY CORPORATION

AUSTRALIA	BELGIUM	U. K.
91 KURRAJONG ROAD MOUNT DRUITT, N.S.W. 2770	PARC INDUSTRIEL DE NANINNE 5140 NANINNE	BILSTON ROAD WEDNESBURY WEST MIDLANDS WS10 7JN

04/83-TM

Printed in U.S.A.